

35장. 스프레드 문법

1. **스프레드 문법**(spread syntax, 전개 문법) ...은 하나로 뭉쳐 있는 여러 값들의 집합을 펼쳐서 개별적인 값들의 목록으로 만듦.
2. 스프레드 문법 사용할 수 있는 대상은 for ... of 문으로 순회할 수 있는 이터러블에 한정됨.
ex) Array, String, Map, Set, DOM 컬렉션 등
3. 스프레드 문법의 결과는 값이 아님 → 스프레드 문법의 결과는 변수에 할당할 수 X

35.1 함수 호출문의 인수 목록에서 사용하는 경우

1. [예제 35-03]

```
const arr = [1, 2, 3];  
  
// 배열 arr의 요소 중에서 최대값을 구하기 위해 Math.max 사용  
const max = Math.max(arr); // NaN
```

✖ Math.max 메서드는 매개변수 개수를 확정할 수 없는 가변 인자 함수

- 숫자가 아닌 배열을 인수로 전달하면 NaN 반환


➡ 이 같은 문제를 해결하기 위해,

배열을 펼쳐서 요소들을 개별적인 값들의 목록화 후, Math.max 메서드의 인수로 전달해야 함.

2. 스프레드 문법 제공되기 이전에는, Function.prototype.apply 사용했음.

3. [예제 35-07]

```
const arr = [1, 2, 3];  
  
// 스프레드 문법 사용해서 배열 arr을 1, 2, 3으로 펼쳐서 전달  
const max = Math.max(...arr); // 3
```

- 스프레드 문법 사용하면 더 간결하고 가독성 

4. Rest 파라미터와 형태가 동일하여 혼동할 수 있으므로 주의 !

- Rest 파라미터 : 함수에 전달된 인수들의 목록을 배열로 전달받기 위해 이름 앞에 ... 붙이는 것.

- 스프레드 문법 : 여러개 값이 하나로 뭉쳐있는 배열과 같은 이터러블을 펼쳐서 개별적인 값들의 목록을 만드는 것.

⇒ 서로 반대의 개념

35.2 배열 리터럴 내부에서 사용하는 경우

35.2.1 concat

1. 2개의 배열을, 1개로 결합하고 싶은 경우 concat 메서드

```
// ES5에서는 concat 메서드 활용
var arr = [1, 2].concat([3, 4]);
console.log(arr); // [1, 2, 3, 4]
```

↓

```
// ES6
const arr = [...[1, 2], ...[3, 4]];
console.log(arr); // [1, 2, 3, 4]
```

35.2.2 splice

```
var arr1 = [1, 4];
var arr2 = [2, 3];

arr1.splice(1, 0, arr2);
console.log(arr1); // [1, [2, 3], 4]

// 배열 자체가 추가되어버림.
```

```
var arr1 = [1, 4];
var arr2 = [2, 3];

// 따라서 Function.prototype.apply 메서드 사용
Array.prototype.splice.apply(arr1, [1, 0].concat(arr2));
console.log(arr1); // [1, 2, 3, 4]
```

```
// 스프레드 문법 사용하면 간결하고 가독성 좋게 표현 가능
const arr1 = [1, 4];
const arr2 = [2, 3];
```

```
arr1.splice(1, 0, ...arr2);
console.log(arr1); // [1, 2, 3, 4]
```

35.2.3 배열 복사

```
// ES5에서는 slice 메서드 활용
var origin = [1, 2];
var copy = origin.slice();

console.log(copy); // [1, 2]
console.log(copy === origin); // false
```

↓

```
// ES6
const origin = [1, 2];
const copy = [...origin];

console.log(copy); // [1, 2]
console.log(copy === origin); // false
```

- 이때 각 요소를 얕은 복사(shallow copy) 함. slice 메서드도 마찬가지

35.2.4 이터러블을 배열로 변환

1. ES5에서 이터러블을 배열로 변환하려면 apply 메서드 or call 메서드 사용하여 slice 호출
 - 이터러블이 아닌 유사 배열 객체도 이 방법으로 가능.
2. 스프레드 문법 사용하면 좀 더 간편하게 이터러블을 배열로 변환할 수 있다.

```
function sum(){
  return [...arguments].reduce((pre, cur) => pre + cur, 0);
}

console.log(sum(1, 2, 3)); // 6
```

3. 더 나은 방법은 Rest 파라미터를 사용하는 것이다.

```
const arrayLike = {
  0: 1,
  1: 2,
  2: 3,
```

```

    length: 3
  }

const arr = [...arrayLike]; // TypeError

```

- 단, 이터러블이 아닌 유사 배열 객체는 스프레드 문법의 대상이 될 수 없다.
→ ES6에서 도입된 Array.from 메서드를 사용

35.3 객체 리터럴 내부에서 사용하는 경우

1. 스프레드 프로퍼티가 제안되기 이전에는 ES6에서 도입된 `Object.assign` 메서드를 사용하여 여러 개의 객체를 병합하거나 특정 프로퍼티를 변경 또는 추가했다.
2. 스프레드 프로퍼티는 `Object.assign` 메서드를 대체할 수 있는 간편한 문법이다.

```

// 객체 병합. 프로퍼티가 중복되는 경우 뒤에 위치한 프로퍼티가 우선권 갖는다.
const merged = { ...{ x: 1, y: 2 }, ...{ y: 10, z: 3 } };
console.log(merged); // { x: 1, y: 10, z: 3 };

// 특정 프로퍼티 변경
const changed = { ...{ x: 1, y: 2 }, y: 200 };
console.log(changed); // { x: 1, y: 200 }

// 프로퍼티 추가
const added = { ...{ x: 1, y: 2 }, z: 0 };
console.log(changed); // { x: 1, y: 2, z: 0 }

```